ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

Neural Architecture Search for Single Image Super-resolution

Master Thesis

Hüseyin Ziya İmamoğlu
Department of Electrical Enginering and Infotmation Technology

Advisors: Yawei Li, Prof. Dr. Radu Timofte
Supervisor: Prof. Dr. Luc Van Gool

September 20, 2021

Abstract

Neural Architecture Search (NAS) aims to automatically learn neural network topologies for given datasets and tasks. Single image super-resolution is one of such tasks where NAS approaches have been used. The work of Gao et al. explored the single path one-shot NAS paradigm with uniform sampling and showed that it provided state of art performance on Imagenet dataset for a classification task. In Lightweight Image Super-Resolution with Information Multi-distillation Network, Hui et. al. proposed a lightweight network, which showed increased performance on super-resolution tasks with less number of parameters compared to the previous works. In order to improve the performance of existing super-resolution networks and build a successful NAS paradigm for single-image super-resolution, we combined these two different works. We created a supernet built on the single one-shot NAS with uniform sampling methodology; with a search space based on lightweight distillation network. The search space is also inspired by Res2Net. Unfortunately, we failed to beat the baselines although our found networks showed comparable performance, showing the premise of the idea and the possibility of further research.

Acknowledgements

I would like to thank Dr. Radu Timofte and Yawei Li for their consistent support and feedback during the project. They were very helpful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Neural Architecture Search (NAS) is an emerging area of deep learning that primarily deals with finding neural network topologies automatically. In other words, the main aim is not just to train a given neural network architecture to find the optimal weights but also to learn the optimal architecture of the network itself for a given tasks. Despite the existence of earlier scientific works [23], the interest in the field has increased rapidly in recent years with novel works like [28], [3]. Different methodologies have been implemented for search spaces, ones which are based on discrete search spaces and reinforcement learning ended up with long runtimes. Improvements have been proposed on top of these works to both increase the efficiency of NAS algorithms and to use them for different scenarios. One such improvement was Single Path One-Shot Neural Architecture Search with Uniform Sampling [10]. This work, which was based on the one-shot NAS paradigm, successfully addressed the shortcomings of one-shot approaches, giving state-of-art results on Imagenet dataset. The paradigm proposed tuned out to be efficient and flexible, showing premise for further use. Our work is based on this one-shot paradigm as we tried to extend this framework to single image super-resolution. We based our super-network on the work of Hui et.al [12], trying to increase the performance of the multi-distillation blocks outlined in the work. For this purpose, we also drew inspiration from Res2Net [8], as the work outlined another building block for incorporating multi-scale information. In the end, we implemented one-shot neural architecture search with uniform sampling for single-image super-resolution with a multi-distillation block. We implemented different supernets and although we fail to beat the baselines, we got comparable results, thus showing the efficacy of our method.

Chapter 2

Related Work

The early works for automatically finding neural network topologies goes back to as early as early 2000s [23]. With the widespread popularity of deep learning and increase in hardware power, different methods have risen for neural architecture search (NAS). In this review, first we will provide an overview of single image super-resolution with a focus on deep learning methodology. Afterwards, we will provide a general overview of NAS methodology with a focus on automated methods for super-resolution. Finally, we will review the one-shot methods and their advantages.

2.1 Single image super-resolution

Single image super-resolution deals with constructing high-resolution (HR) images from the low resolution (LR) images. Low resolution images are the degraded versions of the HR ones and modelled by blurring, down-sampling and corruption processes respectively. This problem is ill-posed as many high resolution representations can exist that can be downsampled to a low resolution image. In order to solve this problem, we model the inverse problem as a Maximum A Posterior (MAP) problem and then solve it. With the advances in deep learning, a convolutional neural network that can directly map the low-resolution images to high resolution images can be created and trained. One such work was the work of Deng et al. which is called SRCNN [6]. This was three layer network that founded a direct relationship between low resolution and high resolution images. Improved networks such as the work of Zhange et. al [27] have been proposed and shown to increase the performance of the network. Kim et al. [14] increased the depth of an SR network to 20, increasing the performance further, outperforming the SRCNN by a significant margin. This was a rather large network and to make the learning process faster, global residual learning with a high initial learning rate was employed. The general trend in these works was to increase the number of convolutions for increasing the PSNR values of the final results. For decreasing the number of parameters, recursive architecture and parameter sharing were utilized in some of the works [16], [24], [11]. On top of that to shorten the time, sub-pixel convolutions were realized in some of the works. Fast SRCNN was one of the examples for this network [7]. A super-resolution network based on a Laplacian pyramid structure was also proposed in the same vein [18]. Another breakthrough was EDSR [20], where authors got rid of the unnecessary modules. These works decreased the number of parameters but required higher number of calculations as they increased the depth to make up for the lost performance. To combat this, Ahn et al. created CARN-M [2], with cascading for a mobile sceneario. This cam at a cost of reduced PSNR. To create a relatively light network Hui et al. proposed a super-resolution network on top of an earlier work [13] which has superior super-resolution performance [12]. In this work, an information multi-distillation block

was proposed where information with different scales have been merged together in the hopes of increasing the network's super-resolution performance. A PSNR score of 32.21 has been reported on Set5, for a scale of 4. Here, we will also entertain another work, namely Res2Net. This is a new backbone architecture for combining multi-scale information [8]. This is also an important work for super-resolution as the authors provide a new backbone for tasks that require multi-scale information which also includes single image super-resolution.

2.2 Neural Architecture Search

Neural Architecture Search (NAS) mainly deals with finding optimal network architectures automatically for a given task and dataset. There are quite a few different methodologies used in NAS, resulting in different search spaces, optimization algorithms and backbones. Some methodologies leverage a discrete search space with reinforcement learning while others might leverage a continuous search space with differentiable loss functions [17]. NAS primarily has two optimization procedures. One is optimising the weights while the other is optimising the architecture. Earlier works did this in a nested manner. Nested optimitation ended up causing high computataional complexity as architectures sampled had to be trained from scratch. With this setup, only small search spaces and datasets were affordable. To combat this, newer search algorithms adopted weight sharing [5]. Some of them even converted the space into continous ones for easier optimization [21]. Here, we will mainly focus on one-shot search methods. One-shot methods consist of training a supernet once and then sampling the final architecture from that supernet, hence the name one-shot [4]. This, in contrast to the most-weight sharing arrangements, doesn't use architecture relaxation, addressing the architecture search problem in a second step. However, the coupling of the supernet weights still remains due to the nature of the optimization issue. The work of Guo et. al. [9] alleviates this issue by using a uniform sampling approach. The approach is state of the art in terms of efficiency, accuracy and training time on ImageNet dataset.

Chapter 3

Proposed Approaches

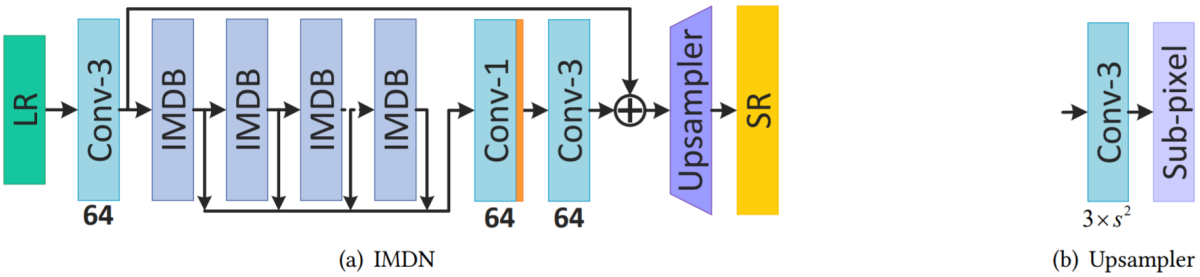
3.1 Preliminaries

3.1.1 Lightweight Image Super-Resolution with Information Multi-distillation Network (IMDN)

We first start with reviewing the main building block of our work which is a lightweight network for single image super-resolution. Hui et al. [12], proposed a neural network where information coming from different scales are progressively refined with contrast-aware channel attention modules on top. The architecture consists of three main building blocks, namely progressive refinement module, upsampler and contrast aware channel attention module. These building blocks form the final network for single image super-resolution. The architecture is depicted in figure 3.1, together with the upsampling module.

The network takes an downsampled image and upscales it for a given upscaling factor in a $I^{SR} = H_{IMDN}(I^{LR})$ fashion. Here, I^{HR} and I^{SR} denotes the high resolution image and the low resolution image respectively while H_{IMDN} denotes the network. There is also a different version of the network created for any upsampling rate but we are nor going to review it here as our work is mainly based on initial network. The network optimizes the loss function $L(\theta) = \frac{1}{N}\sum_{i=1}^{N} ||H_{IMDN}(I_i^{LR} - I_i^{HR}||_1$. As seen from figure 3.1, the network consists of IMDB blocks. These blocks are made up of progressive refinement module and channel attention module. Below we provide detailed descriptions for each module.

The progressive refinement module first extracts features by a 3x3 convolutional layer and then inputs these features through subsequent refinement steps. The idea is to keep some features for the final feature map (refinement) while sending others to the succeeding convolutions for further distillation. This enables important features in a distilled fashion and enabling the fusion of information coming with different focuses



(a) IMDN (b) Upsampler

Figure 3.1: The architecture of information multi distillation network (IMDN) [12]

on the image. A formal description is provided below in equation 3.1. Each convolution is followed by a leaky relu activation function.

$$\begin{aligned} F^n_{refined_1}, F^n_{coarse_1} &= Split^n_1(CL^n_1(F^n_{in})), \\ F^n_{refined_2}, F^n_{coarse_2} &= Split^n_2(CL^n_2(F^n_{coarse_1})), \\ F^n_{refined_3}, F^n_{coarse_3} &= Split^n_3(CL^n_3(F^n_{coarse_2})), \\ F^n_{refined_4} &= CL^n_4(F^n_{coarse_3}). \end{aligned} \tag{3.1}$$

Contrast aware attention layer is an extension on channel attention currently employed in squeeze-and-excitation module (SE). The key idea, which is the same for all attention modules is to highlight the high-value areas. This attention is special to low-level vision and implemented by replacing global average pooling with the summation of standard deviation and mean. If we denote the input as $X = x_1, ..., x_c, , , x_C$ with C feature maps with a spatial size of H x W, the contrast information will be calculated by the equation 3.2 provided below.

$$\begin{aligned} z_c &= H_{GC}(x_c) \\ &= \sqrt{\frac{1}{HW}\sum_{(i,j)\in x_c}(x_c^{i,j} - \frac{1}{HW}\sum_{(i,j)\in x_c x_c^{i,j}})^2} \\ &\quad + \frac{1}{HW}\sum_{(i,j)\in x_c} x_c^{i,j}. \end{aligned} \tag{3.2}$$

The architecture of the multi-distillation block (IMDB) is provided in figure 3.2 while the architecture for contrast aware channel-attention is provided in figure 3.3.

3.1.2 Single Path One-Shot Neural Architecture Search with Uniform Sampling

We start this section of review by giving a general view of NAS approaches which led to the development single path one-shot neural architecture. As stated in related work part, the main aim of NAS is to find architectures automatically. In this regard, we solve two problems weight optimization and architecture optimization. These problems can be formally stated as

$$\begin{aligned} w_a &= argmin_w L_{train}(N(a, w)) \\ a^* &= argmax_{a\in A} ACC_{val}(N(a, w_a)) \end{aligned} \tag{3.3}$$

where L_{train} is the loss function on the training set and ACC_{val} is the validation accuracy. The problem considered here is classification.

Initial approaches, solved this problems in a nested manner, i.e. sampling architectures from A and training them from scratch. To solve this problem, weight sharing is introduced where a supernet that consists of the search space was created and trained. This supernet is optimized and then the best architecture is sampled from this network. This approach came with the problems of weight coupling during optimization and joint optimization leading to some parts of the network being trained better than the others. To further solve this one-shot methods were introduced. In this paradigm, the supernet is trained once without any architecture relaxation and the best network is sampled from this supernet. In other words, the training

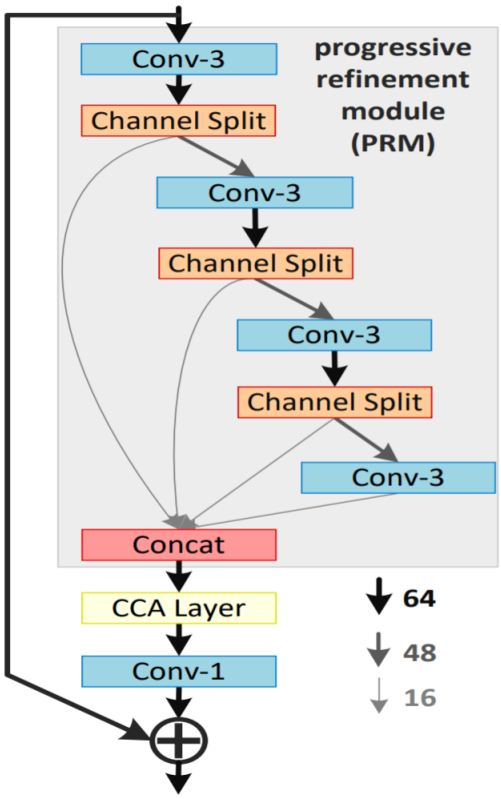


Figure 3.2: The architecture of information multi distillation block (IMDB) [12]

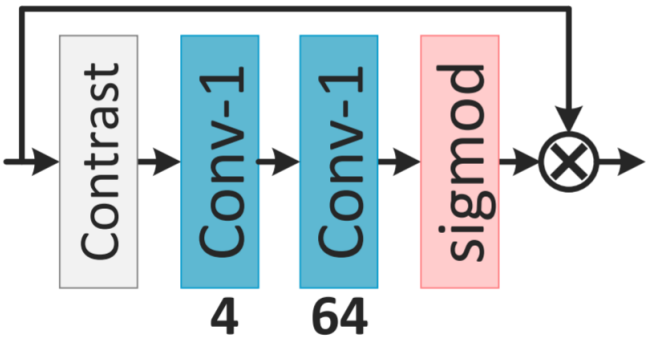


Figure 3.3: The architecture of contract-aware channel attention (CCA) [12]

phase and the search phase are decoupled from each other. This made the process very efficient, flexible and feasible. One-shot approaches tried to solve the problem of weight-sharing by using a drop-out strategy [10].

In their work, Guo et. al. observed that solving the problem of weight-coupling with a path dropout strategy yield results that are very sensitive to the dropout rate parameter. Thus, they proposed a single path supernet architecture with uniform sampling to address the issues outlined in the previous paragraph. In this paradigm, each network is a single path in the supernet that is being optimized and one is chosen uniformly during the training procedure. This effectively addresses the problems of weight-coupling and unequal training of network. The trained supernet is then evaluated with an evolutionary algorithm and the best performing model is trained one more time in a fine-tuning fashion. The algorithm for evolutionary search is outlined in algorithm 1 [10].

Algorithm 1: Evolutionary Architecture Search

Input: *supernet weights* $\mathrm{W}_{\mathcal{A}}$*, population size P, architecture constraints* $\mathcal{C}$*, max iteration* $\mathcal{T}$*, validation dataset* D_{val}
Output: *the architecture with highest validation accuracy under architecture constraints*
$\mathrm{P}_0 := Initialize\_population(P, \mathcal{C})$; $\mathrm{Topk} := \emptyset$;
$n := P/2$; Crossover number
$m := P/2$; Mutation number
$prob := 0.1$; Mutation probability
for $i = 1 : \mathcal{T}$ **do**
  $\mathrm{ACC}_{i-1} := Inference(\mathrm{W}_{\mathcal{A}}, \mathrm{D}_{val}, \mathrm{P}_{i-1})$;
  $\mathrm{Topk} := Update\_Topk(\mathrm{Topk}, \mathrm{P}_{i-1}, \mathrm{ACC}_{i-1})$;
  $\mathrm{P}_{crossover} := Crossover(\mathrm{Topk}, n, \mathcal{C})$;
  $\mathrm{P}_{mutation} := Mutation(\mathrm{Topk}, m, prob, \mathcal{C})$;
  $\mathrm{P}_i := \mathrm{P}_{crossover} \cup \mathrm{P}_{mutation}$;
end
Return the architecture with highest accuracy in Topk;

We did not provide the details of the architecture here as this work was implemented for a classification task on ImageNet dataset.

3.1.3 Res2Net

Res2Net is a backbone proposed by Gao et. al. [8] to extract multiscale information and to represent this information better in convolutional neural networks. The work proposes to do the multi-scaling at a more granular level and to have multiple receptive fields available. To achieve this, the authors replaced the 3x3 convolutions with n channels by smaller filters with w channels. The feature maps are then concatenated in the end. The module compared with the bottleneck block in the architecture is provided in figure 3.4. This led to a performance increase when used with state-of-art classification networks on ImageNet. We are not going to go in further detail in reviewing this paper as we use the Res2Net building block as an inspiration in our custom supernet design.

3.2 Building the Supernet

In order to effectively search for a network using the one-shot paradigm proposed in [10], we first have to build a supernet. Over the course of our work, we built a couple of different supernet in the process of

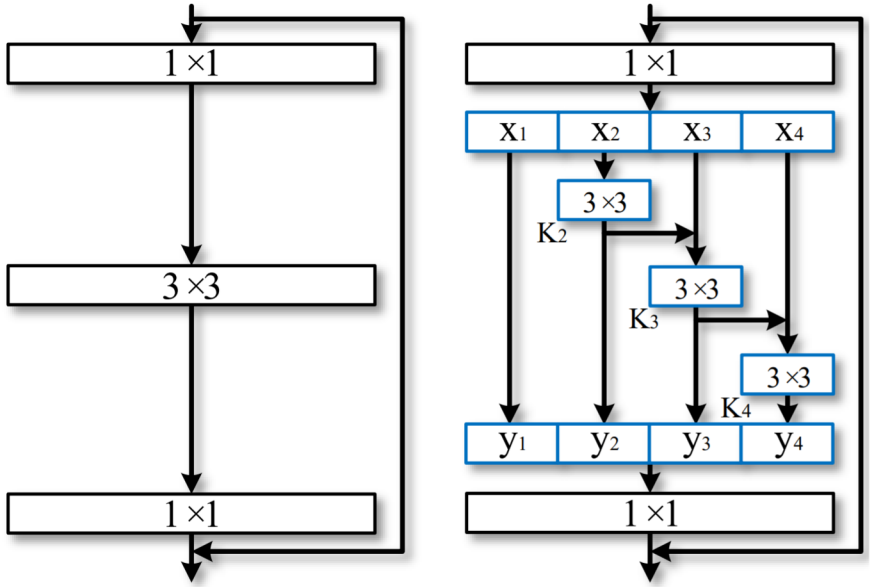


Figure 3.4: Res2Net building block in comparison with the bottleneck block [8]

increasing the performance. Here, we are going to provide an overview of these nets. The nets all have the property of collapsing into single path networks with sampling. One thing that we should talk about is also the minor changes that we implemented in our networks. More details will be provided in the next chapter, where we also discuss the results of our work.

Before going further, we should also give some motivation for our reasoning. We started with a relatively small supernet and enhanced the network both by deepening it and also implementing different connections.

3.2.1 Initial Supernet and extensions

Our initial supernet is based on the IMDB which is depicted in 3.2. However, since our aim here is to establish the proof-of-concept of our architecture, we started with three restrictions on IMDB, namely;

1. We decreased the number of stages (number of x3 convolutions two three) in the initial network.
2. We necessitated that the following a convolutions there should be a channel-split
3. The general structure is preserved as we still use concatenation as the final operation

The first two restrictions are gradually lifted. We added convolutions to the skip-connection layers inspired by [8]. The first architecture is provided in figure 3.5.

As seen from figure 3.5, the supernet encapsulates 5 different paths which it can collapse into during training. By altering the number of channel splits and convolutions in the architecture, we are in a way searching for the best combination of the operations and best number of channels. With adding convolutions to the skip connections, we extract further features. In the initial version of our network, we whether have skip-connections or convolutions for the operations outlined as the connections to concatenations from split

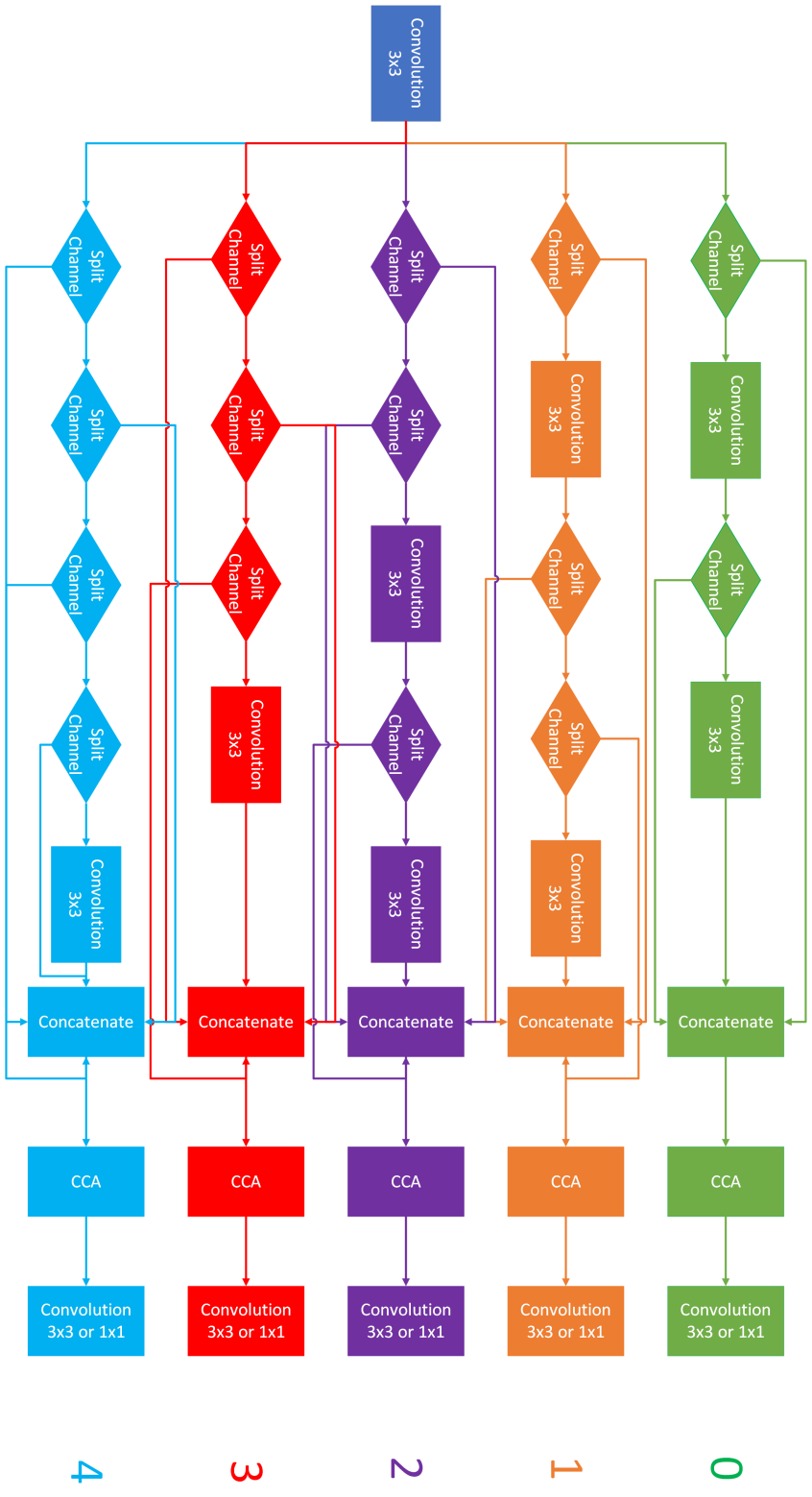


Figure 3.5: Our initial supernetwork with each architecture colored different with numbering

channels 3.5. however we also extended this network to choose between 3x3 or bypass, effectively having 20 choices.

Another thing in the new network is the final concatenation. In our early experiments, we concatenated them directly without decreasing the number of channels in the last convolution to account for 64 channel size. This increased the number of parameters significantly so we changed the last convolution in our network to compensate. This increased the number of parameters and thus we also created a version with less number of channels in skip-connections by altering the splits. We will provide further implementation details in the results section.

In addition, in the initial network the first convolution in IMDB and the CCA layer with the last convolution are not shared between the different paths. We also built networks with this structure shared. This pretty much meant that the final layers of each net is shared.

3.2.2 Dynamic Supernet

Dynamic supernet is the version of the network in section 3.2.1 where we get rid of the first two of our restrictions. This achitecture can be scaled to an arbitrary number of convolutions and thus arbitrary number of supernetworks. It's basically a generalized version of the initial supernet outlined in 3.2.1. This scales with the power of twos as for a network of 3 number of stages we have $2^3 = 8$ number of different paths that a network can take. Figure 3.6 shows the different paths for the dynamic architecture with 3 number of stages. Further details are provided in the next sections.

Figure 3.6: Our dynamic supernetwork for 3 number of stages with each architecture colored different with numbering

Chapter 4

Experiments and Results

4.1 Dataset

There are 3 different datasets used in the experiments in this work. We use the DIV2K [1] dataset for training the neural networks. Set5 is used for validating the networks while Set14, Urban100 and BSD100 are used for testing purposes.

4.2 Initial experiments with IMDN and Establishing a Baseline

These experiments were done with using the training set as DIV2K and using the exact procedure outlined by [12]. The network is trained for a full 1000 epochs with an Adam optimizer with a momentum parameter of $\beta_1 = 0.9$. The iniial learning rate is $2x10^{-4}$ and it's halved at every $2x10^5$ iterations. The number of IMDB units is 6. We ran the experiment for different scalings of x4, x3 and x2. The results are provided in table 4.2. The results reported in the paper are provided in table 4.1. The values in bold show the best results while the underlined values are the second best results. The table also provides the number of parameters for each network. The table is directly taken from [12].

As one can see, we did not quite manage to get the exact results in the paper although we have used the exact methodology and the same dataset. However, we though that the results were close enough, especially for the scale of x3 and as we were trying the achieve comparable results, we decided to keep on working and tried to see whether we could improve on these baselines. Since the original work produced put an emphasis on the lightweight structure of the model, we will also replicate this emphasis in our work throughout the report and provide the number of parameters for each network that we created. We also restricted our work for x4 upscaling as it was the fastest setting for evaluation. This, we believed, would lead us to more experiments given a time frame, meaning that we would have more time to experiment with different ideas. Thus, from now on, all the architectures that are trained are for x4 upscaling.

After having the initial model, we decided to add some new blocks and then play with the model parameters to get a sense of how to improve the building block. This was done in order to understand the limitations and strengths of the block as well as to inquire about possible paths that might lead to a better supernet. The results are provided in table 4.3. Before proceeding further with the results of the network, we should explain what those different directions mean. We wanted to first test the kernel size in our network as kernel size is something NAS algorithms usually search for. For this purpose we devised two experiments. We first increased the number of kernels to five and then use selective kernel [19] paradigm with different settings. Selective kernel operations is an automated convolution block where a combination of different

Table 4.1: PSNR and SSIM results for different scales from [12]

Method	Scale	Params	Set5	Set14	BSD100	Urban100	Manga109
			PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM
Bicubic	×2	-	33.66 / 0.9299	30.24 / 0.8688	29.56 / 0.8431	26.88 / 0.8403	30.80 / 0.9339
SRCNN [6]		8K	36.66 / 0.9542	32.45 / 0.9067	31.36 / 0.8879	29.50 / 0.8946	35.60 / 0.9663
FSRCNN [7]		13K	37.00 / 0.9558	32.63 / 0.9088	31.53 / 0.8920	29.88 / 0.9020	36.67 / 0.9710
VDSR [15]		666K	37.53 / 0.9587	33.03 / 0.9124	31.90 / 0.8960	30.76 / 0.9140	37.22 / 0.9750
DRCN [16]		1,774K	37.63 / 0.9588	33.04 / 0.9118	31.85 / 0.8942	30.75 / 0.9133	37.55 / 0.9732
LapSRN [18]		251K	37.52 / 0.9591	32.99 / 0.9124	31.80 / 0.8952	30.41 / 0.9103	37.27 / 0.9740
DRRN [11]		298K	37.74 / 0.9591	33.23 / 0.9136	32.05 / 0.8973	31.23 / 0.9188	37.88 / 0.9749
MemNet [24]		678K	37.78 / 0.9597	33.28 / 0.9142	32.08 / 0.8978	31.31 / 0.9195	37.72 / 0.9740
IDN [13]		553K	37.83 / 0.9600	33.30 / 0.9148	32.08 / 0.8985	31.27 / 0.9196	38.01 / 0.9749
EDSR-baseline [20]		1,370K	37.99 / 0.9604	33.57 / 0.9175	32.16 / 0.8994	31.98 / 0.9272	38.54 / 0.9769
SRMDNF [26]		1,511K	37.79 / 0.9601	33.32 / 0.9159	32.05 / 0.8985	31.33 / 0.9204	38.07 / 0.9761
CARN [2]		1,592K	37.76 / 0.9590	33.52 / 0.9166	32.09 / 0.8978	31.92 / 0.9256	38.36 / 0.9765
IMDN [12]		694K	**38.00 / 0.9605**	**33.63 / 0.9177**	**32.19 / 0.8996**	**32.17 / 0.9283**	**38.88 / 0.9774**
Bicubic	×3	-	30.39 / 0.8682	27.55 / 0.7742	27.21 / 0.7385	24.46 / 0.7349	26.95 / 0.8556
SRCNN [6]		8K	32.75 / 0.9090	29.30 / 0.8215	28.41 / 0.7863	26.24 / 0.7989	30.48 / 0.9117
FSRCNN [7]		13K	33.18 / 0.9140	29.37 / 0.8240	28.53 / 0.7910	26.43 / 0.8080	31.10 / 0.9210
VDSR [15]		666K	33.66 / 0.9213	29.77 / 0.8314	28.82 / 0.7976	27.14 / 0.8279	32.01 / 0.9340
DRCN [16]		1,774K	33.82 / 0.9226	29.76 / 0.8311	28.80 / 0.7963	27.15 / 0.8276	32.24 / 0.9343
LapSRN [18]		502K	33.81 / 0.9220	29.79 / 0.8325	28.82 / 0.7980	27.07 / 0.8275	32.21 / 0.9350
DRRN [11]		298K	34.03 / 0.9244	29.96 / 0.8349	28.95 / 0.8004	27.53 / 0.8378	32.71 / 0.9379
MemNet [24]		678K	34.09 / 0.9248	30.00 / 0.8350	28.96 / 0.8001	27.56 / 0.8376	32.51 / 0.9369
IDN [13]		553K	34.11 / 0.9253	29.99 / 0.8354	28.95 / 0.8013	27.42 / 0.8359	32.71 / 0.9381
EDSR-baseline [20]		1,555K	**34.37 / 0.9270**	30.28 / **0.8417**	**29.09 / 0.8052**	28.15 / **0.8527**	33.45 / 0.9439
SRMDNF [26]		1,528K	34.12 / 0.9254	30.04 / 0.8382	28.97 / 0.8025	27.57 / 0.8398	33.00 / 0.9403
CARN [2]		1,592K	34.29 / 0.9255	30.29 / 0.8407	29.06 / 0.8034	28.06 / 0.8493	33.50 / 0.9440
IMDN [12]		703K	34.36 / **0.9270**	**30.32 / 0.8417**	**29.09** / 0.8046	**28.17** / 0.8519	**33.61 / 0.9445**
Bicubic	×4	-	28.42 / 0.8104	26.00 / 0.7027	25.96 / 0.6675	23.14 / 0.6577	24.89 / 0.7866
SRCNN [6]		8K	30.48 / 0.8628	27.50 / 0.7513	26.90 / 0.7101	24.52 / 0.7221	27.58 / 0.8555
FSRCNN [7]		13K	30.72 / 0.8660	27.61 / 0.7550	26.98 / 0.7150	24.62 / 0.7280	27.90 / 0.8610
VDSR [15]		666K	31.35 / 0.8838	28.01 / 0.7674	27.29 / 0.7251	25.18 / 0.7524	28.83 / 0.8870
DRCN [16]		1,774K	31.53 / 0.8854	28.02 / 0.7670	27.23 / 0.7233	25.14 / 0.7510	28.93 / 0.8854
LapSRN [18]		502K	31.54 / 0.8852	28.09 / 0.7700	27.32 / 0.7275	25.21 / 0.7562	29.09 / 0.8900
DRRN [11]		298K	31.68 / 0.8888	28.21 / 0.7720	27.38 / 0.7284	25.44 / 0.7638	29.45 / 0.8946
MemNet [24]		678K	31.74 / 0.8893	28.26 / 0.7723	27.40 / 0.7281	25.50 / 0.7630	29.42 / 0.8942
IDN [13]		553K	31.82 / 0.8903	28.25 / 0.7730	27.41 / 0.7297	25.41 / 0.7632	29.41 / 0.8942
EDSR-baseline [20]		1,518K	32.09 / 0.8938	28.58 / **0.7813**	27.57 / **0.7357**	26.04 / **0.7849**	30.35 / 0.9067
SRMDNF [26]		1,552K	31.96 / 0.8925	28.35 / 0.7787	27.49 / 0.7337	25.68 / 0.7731	30.09 / 0.9024
CARN [2]		1,592K	32.13 / 0.8937	**28.60** / 0.7806	**27.58** / 0.7349	**26.07** / 0.7837	**30.47 / 0.9084**
IMDN [12]		715K	**32.21 / 0.8948**	28.58 / 0.7811	27.56 / 0.7353	26.04 / 0.7838	30.45 / 0.9075

Table 4.2: Reproduced PSNR for different scales together with reported results

Depth	**Scale**	**Distillation Rate**	**Set 5 PSNR (Trained from scratch)**	**Set 5 PSNR (Claimed)**
6	x2	0.250	37.87	38.00
6	x3	0.250	34.34	34.36
6	x3	0.250	32.14	32.21

Table 4.3: Preliminary results

Network Type	Kernel Size	Number of stages	Distillation rate	Set5 PSNR	Parameter Size
Vanilla	5x5	4	0.25	32.1433	2333436
Vanilla	3x3	4	0.25	32.1206	715000
Vanilla	3x3	3	0.3	32.0012	637848
Vanilla	3x3	4	0.25	32.0625	839880
Vanilla	3x3	4	0.125	32.027	908028
Vanilla with Selective Kernel	(1x1, 3x3)	4	0.25	32.0608	-
Vanilla with Selective Kernel	(3x3, 5x5)	4	0.25	32.08	-

size convolutions can be combined and selected. We also wanted to see the effect of the splits as this was something that NAS algorithms also focus on as splits determine the channel size inputted into the convolutions. We used different distillation rates, i.e. the multiplier for determining the number of channels distilled to the final concatenation in IMDB unit. The last thing that we tested for is the number of stage, i.e. the number of convolutions used in the IMDB unit. We did this because we wanted to see how the depth of the network affects the performance.

Since this was to get a sense of the architecture, we only used the Set5 dataset for evaluations. The ideas was to see which directions which we can extend the network into. The early experiments did not give us the results that we were hoping for however this was a very important result as it showed us two very important things about the network itself. One is that increasing kernel size increases the performance although with a huge number of increase in parameter size. The second is that different distillation rates as shown effect the performance, albeit not very significantly, giving us hope that this direction might yield better results without increasing the number of paramaters too much. Different number of stages are also checked here with using 3 stages. Although the performance decreased, the number of parameters decreased as well, which gave use hope as we might be able to increase the performance of the network with NAS with 3 stages. We also observed variability of the results as the average PSNR value fluctuated for differing vanilla trials for scaling x4.

4.3 Initial Supernet and Extensions

Here, we provide the results for our initial supernetwork which we outlined its building block in figure 3.5. We have 6 blocks, however for this particular experiment we use the Set5 as the validation set and use Set14 and BSD100 for the final test. The training set is DIV2K. Number of epochs is increased to 2000 and different learning rate schedulers are used for some of the experiments (explicitly mentioned). After the supernet is trained, just as outlined in [12], we again train the found network using the same training

Table 4.4: Initial Supernet results and extensions

Network type	Depth	LR Scheduler	Distillation rate	Val PSNR(Set 5)	Set 14 PSNR	BSD100 PSNR	Urban100 PSNR	Parameter Size	Chosen Path for Each Block
SuperNet	6	Step	0.125	32.08	28.49	27.51	25.91	922856	3,2,4,4,4,2
SuperNet	8	Step	0.125	32.16	28.54	27.53	26.02	1203921	4,3,4,2,2,3,4,2
SuperNet (Reduced)	8	Step	0.250	32.14	28.50	27.50	25.92	1047062	3,3,4,4,4,4,3,4
SuperNet (Reduced)	10	Step	0.250	32.21	28.55	27.55	26.03	1369387	-
SuperNet (Reduced)	6	Cosine	0.250	32.09	28.47	27.47	25.87	772368	3,4,3,4,4,4
SuperNet (Reduced - Shared)	**6**	**Cosine**	**0.250**	**32.09**	**28.50**	**27.50**	**25.91**	**755508**	3,4,2,2,4,4
SuperNet (Reduced)	8	Step	0.250	32.16	28.55	27.52	26.02	1005212	3,3,4,4,4,4,3,4
SuperNet (Reduced)	**8**	**Cosine**	**0.250**	**32.21**	**28.57**	**27.54**	**26.06**	**1005212**	**3,3,4,4,4,4,3,4**
SuperNet (Reduced - Shared-Conv1)	6	Cosine	0.250	32.04	28.47	27.49	25.86	556296	4,2,3,3,3,4
Vanilla	6	Cosine	0.250	32.15	28.55	27.52	26.00	712000	-
Vanilla	8	Cosine	0.250	32.19	28.57	27.54	26.06	931408	-
Vanilla	10	Cosine	0.250	32.25	28.62	27.57	26.15	1147640	-
Trilevel NAS [25] (Claimed)	-	-	-	31.62	28.26	-	-	510000	-

parameters. The results for this training scheme and the network are provided in table 4.4.

Before providing any commentary on the results, we should provide the necessary information about the architectures. Vanilla stands for the IMDN architecture with only, with no training alterations except for increasing the number of epochs to 2000 for better comparison. For some trials, the learning-rate scheduling was changed to cosine-annealing learning rate scheduler [22] as it was found to be better for the fine-tuning phase. Each experiment with the SuperNet label, denotes a process of first training the supernet, then searching on the supernet using the evolutionary search algorithm outlined in 1 and then finally fine-tuning the network. The LR scheduler is only changed for the fine-tuning part. The maximum number of iterations is 10 for the each experiment for the scheduler. Distillation rate is the distillation for each channel-split operations and the depth is the number of modified IMDB units that we used (each unit has 5 different paths in them as outlined in figure 3.5). The paths chosen for each block are also shown. Shared stands for the architecture in which initial convolution coupled with the last convolution and the CCA layers are shared between the different paths. The reduced paradigm changes the number of the final channels in the network before concatenation. So the final convolution before concatenation in the IMDB paradigm has an input of 16 channels and also has an output of 16 channels. This makes sense here as the splits are constant there are three splits and four different channel representation are concatenated as outlined if figure 3.2. When we build the supernet, this paradigm is of course broken so we had to find a way to make sure that the number of channels leaving the architecture is 64 so that the final summation can take place after each successive IMDB. In our first implementation, we let the last convolution before the concatenation to have same number of input and output channels while doing the reduction with the last convolution. In the architecture that we defined as reduced, we did this mapping in the last convolution before concatenation rather than in the

last convolution. This is important as this provided similar performance with smaller number of parameters, which also can be seen from table 4.4.

When we take a look at the results of our initial network, they are disappointing. We expected to beat the baselines with the network that we custom-designed however the best results that we obtained are only comparable to the baselines. This is a disappointing result, however one should also see that there's some merit in out methodology. Our process of train-search-train leads to different architectures (different from the IMDN as well) that can compete with the baselines in the best cases. These means that our methodology shows premise as we are not just collapsing into IMDN, which is a possibility in our architectures. The second important thing is the differing sizes for networks with similar depths. We can actually limit the number of parameters or flops during our search. We did not do that here to see the unconstrained version of our network, however this is doable and another upshot of our work. This means one can design custom neural networks with different requirements with this framework for super-resolution tasks. In addition, our search progress also outlined a network with very small number of parameters. This network managed to outperform some of the previous works with comparable parameter size. And one important thing that we should also discuss here is the effect of sharing on the network. By keeping the initial convolutions and the final ones constant between different paths sampled from the supernet distillation blocks, we managed to increase the performance of our network with smaller number of parameters. This is a contribution of ours to the proposed one-shot paradigm as this is not outlined in the original paper. However, most probably, this is a domain-specific setting being valid for this particular design of the network for single image super-resolution.

Another thing that we should pay attention to is the found architecture of the networks. Here, as one can see from figure 3.5, the closest path to the IMDB unit is 4 and it's no consequence that 4 is widely repeated in our results. However, we also see that paths such as 2 and 3 are also chosen. The best performing networks did not choose the first two paths. These results can be read in two different ways. The first one is that the optimum architectures which provide comparable results to the baseline model are similar but not the same as the architecture did not just collapse into the IMDB unit as path 4 is IMDB with three number of stages. The found architectures show variability, and interestingly show repeating patterns in the middle of the network.

4.4 Dynamic Supernet

Here we provide the results with our dynamic network. The dynamic network, as we outlined in section 3.2 has 2^n number of different paths for a given block where n is the number of stages. The results are provided in table 4.5. Here, we should also say that the end concatenations of this network follows the reduced pattern explained above in section 4.3. The network has a depth of 6 and the search is done with the same method outlined also in section 4.3. Number of stages for each network is shown for each network. Here, instead of training only the best network, we also trained the best three networks. We also trained them from scratch together with fine-tuning them. after the training of the supernet is finished.

This network seems to be better than our initial one as the results are a bit better. This supports our initial assumption that increasing the search space will lead to better results. Again they are comparable to the baselines. The interesting thing here is when we train the network from scratch, we sometimes get better results from when we fine-tune the already trained supernet. This is a very important result and shows the viability of our methods as our networks which are different than the initial network leads to comparable results. In addition, this also shows the potential of our methodology as a semi-automated procedure. To elaborate further, we can combine the modules that we found during automated search with the hand-crafted

Table 4.5: Dynamic Supernet results with replicated baselines for differing length

Network type	LR Scheduler	Depth	Distillation rate	Number of Stages	Val PSNR (Set 5)	Val PSNR (Set 5, from scratch)	Set 14 PSNR	BSD100	Urban100	Parameter Size	Chosen Path for Each Block
Dynamic SuperNet (Reduced-Best)	Cosine	6	0.125	3	32.17	32.17	28.58	27.56	26.13	1215187	7,1,7,7,7,7
Dynamic SuperNet (Reduced - Second Best)	Cosine	6	0.125	3	32.18	32.16	28.6	27.56	26.17	1257880	7,5,7,7,7,7
Dynamic SuperNet (Reduced - Third Best)	Cosine	6	0.250	3	32.17	32.19	28.61	27.55	26.15	1159811	7,1,7,7,5,7
Dynamic SuperNet (Best)	Cosine	6	0.250	4	32.24	-	28.55	27.56	26.13	14104962	11,7,12,7,13,13
Vanilla	Cosine	6	0.250	4	32.15	-	28.55	27.52	26.00	712000	-
Vanila	Cosine	8	0.250	4	32.19	-	28.57	27.54	26.06	931408	-
Vanilla	Cosine	10	0.250	4	32.25	-	28.62	27.57	26.15	1147640	-

ones and train the new network from scratch. This might yield to better results and opens up further research possibilities into this particular topic.

The found architectures are also very interesting. The first best three models both have the 7th component as their most repeated building block with one or two different paths in between. This is a very significant result as the 7th block includes subsequent convolutions and thus different from the IMDB unit. The detailed summary of the dynamic net showing different paths are provided in Appendix. 7th path in this architecture setting is most convolution-heavy network just like the 4th path in the initial network 3.5. This might be due to the uneven optimization of the network however single-path architecture search with uniform sampling is known to counter this [10]. Thus, this pattern shows the premise of limiting the network as architectures with less number of parameters might not yield this collapse into 7. Another interesting result is the found architecture when we increase the number of stages to 4. The found architecture is vastly different than the IMDN, however it shows tendencies of overfitting to the validation set. This also shows the possible premise of limiting the number of parameters during architecture search. We did not provide results with this angle as our main aim here is to prove the concept of using one-shot learning with uniform sampling in this setting and thus we did not restrict the number of parameters. We wanted to see the architectures that the process yields.

Chapter 5

Conclusion

In this work, our main aim was to implement one-shot NAS with uniform sampling paradigm [10] to single-image super-resolution task. For that purpose, we devised a supernet based on the lightweight multi-distillation network proposed by HUi et. al [12] (IMDN) and inspired by Res2Net [8].

We built different supernetworks as our work progressed. Although different neural blocks were introduced, we decided to built out network on two main operations, channel split and convolution. We built the supernet by redesigning the IMDB units in IMDN in a multi-path structure where sampling a path from these blocks amounted to creating a network. The initial network was a simple one with 5 different paths for each block with three main restrictions, namely; the number of stages remained constant, every convolution was followed by a skip-convolution layer and concatenation is used for the final operation. Later, we relaxed the first two of these operations in dynamic networks and also implemented some structural improvements. Although the results were comparable of our reproduced baselines, we failed to improve on top of IMDN [12].

The best performing model when taken in tandem with the number of parameters was the dynamic model with the number of stages as 3. This particular model provided directly comparable results to the baseline when we increased the depth of the network to 8. This is an important results and directly affected by the fact that the network in study was designed to be optimal with 6 units. On top of this, we also found out that the networks with higher number of convolutions are favoured and the researched networks do not collapse into IMDN, which led us to say that our work has merit. In addition, the network was also competitive with different works when we decrease the number of parameters [25].

To sum up, we implemented a NAS algorithm for single-image super-resolution by using IMDN backbone and single path one-shot search with uniform sampling. The algorithm is comparable to the baselines but does not improve the results.

Bibliography

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[2] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network, 2018.

[3] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning, 2017.

[4] Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. Smash: One-shot model architecture search through hypernetworks, 2017.

[5] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware, 2019.

[6] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks, 2015.

[7] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network, 2016.

[8] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2net: A new multi-scale backbone architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(2):652–662, Feb 2021.

[9] Shuhang Gu, Wen Li, Luc Van Gool, and Radu Timofte. Fast image restoration with multi-bin trainable linear units. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[10] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling, 2020.

[11] Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. Deep reinforcement learning with a natural language action space, 2016.

[12] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. *Proceedings of the 27th ACM International Conference on Multimedia*, Oct 2019.

[13] Zheng Hui, Xiumei Wang, and Xinbo Gao. Fast and accurate single image super-resolution via information distillation network, 2018.

[14] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks, 2016.

[15] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks, 2016.

[16] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution, 2016.

[17] George Kyriakides and Konstantinos Margaritis. An introduction to neural architecture search for convolutional networks, 2020.

[18] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Fast and accurate image super-resolution with deep laplacian pyramid networks, 2018.

[19] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks, 2019.

[20] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution, 2017.

[21] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[22] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

[23] Kenneth Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10:99–127, 02 2002.

[24] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration, 2017.

[25] Yan Wu, Zhiwu Huang, Suryansh Kumar, Rhea Sanjay Sukthanker, Radu Timofte, and Luc Van Gool. Trilevel neural architecture search for efficient single image super-resolution, 2021.

[26] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Learning a single convolutional super-resolution network for multiple degradations, 2018.

[27] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks, 2018.

[28] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning, 2017.